VS Code with Source Control

by Bryan A. *CrazyUncleBurton* Thompson

Last Updated 2/21/2026

Overview

Millions of hours are wasted every year installing and configuring unique IDEs and compilers and libraries every time the developer wants to try a new microcontroller platform.

VS Code is an Integrated Development Environment from Microsoft. Source Control is a systematic method of working with an existing project or as part of a team. We maintain backup copies of the project in GitHub as we develop, and multiple people can work on a project at the same time.

Our goal is to create a unified development environment with concurrent versioning which supports thousands of microcontrollers and development boards. VS Code and all the extensions and support apps mentioned in this document are all free downloads.

Prerequisites

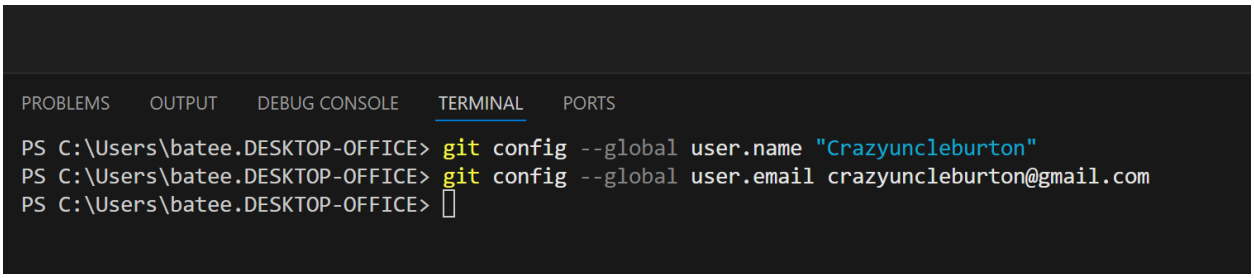
1. A GitHub.com account is required to complete this tutorial. They are free.
2. Admin rights on the computer are needed to install some apps.

Setting Up a New Computer

Note: This section only needs to be done once for each computer.

1. **Browse** to https://git-scm.com/install and download the latest version of Git for the OS the computer is running.
2. **Right-click the git(whatever).exe file** and choose **Run As Administrator**
3. Choose **Yes** to allow the installer to install the program for all users.
4. **Click Next** then **leave the install location alone and click Next**
5. **Click Next** to accept the default choices for components.
6. **Click Next** to accept the Git folder in the Start Menu.
7. **Choose Use Visual Studio Code** as the default editor and **click Next**
8. For adjusting the name of the initial branch, leave the default settings and **click Next**
9. On the "adjusting path" dialog, choose the default and **click Next**
10. Use Bundled OpenSSH, then **click Next**
11. Backend? Choose the default and **click Next**
12. Configuring line endings? Choose the default and **click Next**
13. Configuring Terminal Editor? Choose the default and **click Next**
14. Fast Forward or Merge, then **click Next**
15. Credential helper is Git Credential Manager, then **click Next**
16. **Check Enable File System Caching** then **click Next**
17. Git installs. Finally.
18. Reboot the computer when it finishes.
19. **Browse** to https://code.visualstudio.com/download#
20. **Download VS Code for the OS the computer has installed**. For Windows, use the one marked System instead of the one marked User and instead of the Windows Store version. It will install for all users and give the ability to edit important files.
21. **Right-click the downloaded file and choose Run As Administrator**
22. Give the popup window permission **Yes**
23. **Accept** the license agreement.
24. Choose the default location to install VS Code and **click Next**
25. **Click the Create a Desktop Icon and "Open with Code" options** then **click Next** and then **click Install**
26. Leave Launch Visual Studio Code checked and then **click Finish**
27. Open VS Code.
28. Start a terminal by **clicking Terminal/New Terminal** from the menu.
29. Issue the command **git –version** It should report the version of Git that was just installed.

30. Issue this command, replacing “Your Name” with your GitHub User Name:
    git config --global user.name "Your Name"
31. Issue this command, replacing your.email@example.com with the email address associated with the GitHub account:
    git config --global user.email your.email@example.com



32. Restart the VS Code app

33. From the left-hand toolbar, **click the Extensions icon**:
34. Search **Platform IO** and **install the Platform IO extension**
35. **Trust Platform IO** if a popup box appears.
36. **Click Reload Extensions** after Platform IO installs.
37. From the left-hand toolbar, **click the Extensions icon**:
38. Search for **C/C++**
39. **Install the Microsoft C/C++ Extension** if it’s not already installed. It’s Microsoft, so no trust needed.

40. Restart VS Code

The Platform IO Project

The Platform IO project contains files and folders that with the source code which gets compiled and uploaded to the microcontroller. It also contains files like platformio.ini which tell Platform IO and the compiler what microcontroller we are compiling for. The project can contain one branch or multiple branches for things like past, production and development environments.

When we open the project in VS Code, we are cloning the repository. This is a process which downloads from the cloud repository to the local machine. If there is no existing project to clone, we show the process of creating a new project.

The quickest way to get started in microcontroller development is to find an existing project that works on the microcontroller we're using, and to clone it to the local machine. This project might be in the microcontroller manufacturer's GitHub account (common), or it might be a project by some rando on the Interweb.

Project Structure

.pio/
  build/ - temporary build files, clean to delete these
  libdeps/ - managed (dynamic, changing) libraries go here

.vscode

platformio.ini
README.md – a markdown-style file that creates the docs in your GitHub project.

data/ - Stuff that is stored in the SPIFFS/LittleFS file systems in microcontroller memory.
  image_1.png
  image_2.png

lib/ - These are unmanaged (static) libraries
  unmanaged_library_1/
    unmanaged_library_1.h
    unmanaged_library_1.c

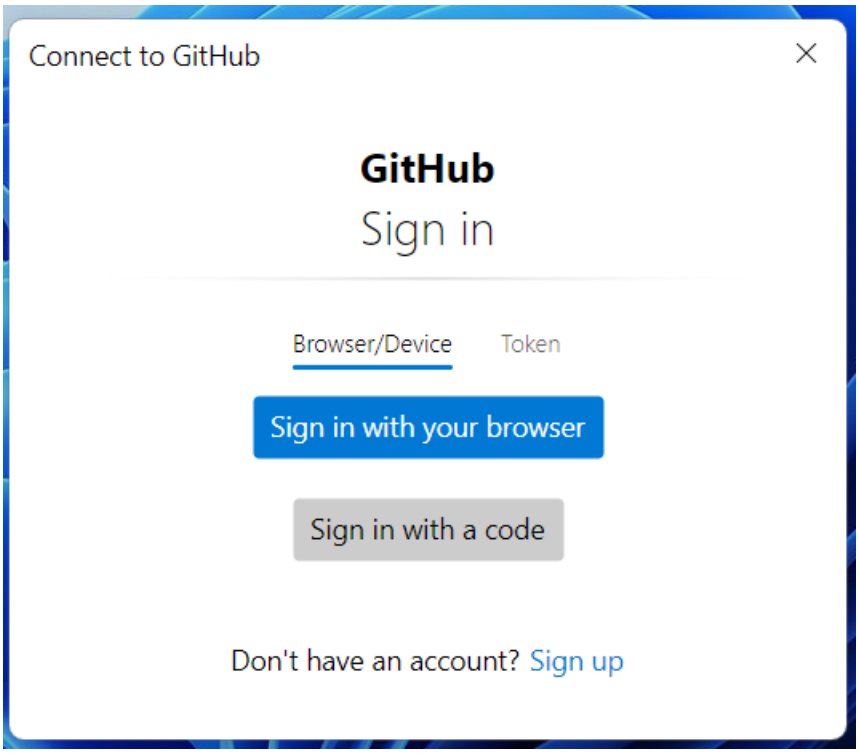
include/
  pin_config.h – a good place to put mapping of variable names to specific pins on the microcontroller. Sometimes supplied by the manufacturer.

src/
  main.cpp – Arduino /ino style file which contains the starting point for your source, or maybe all of it.

Opening an Existing Project in VS Code

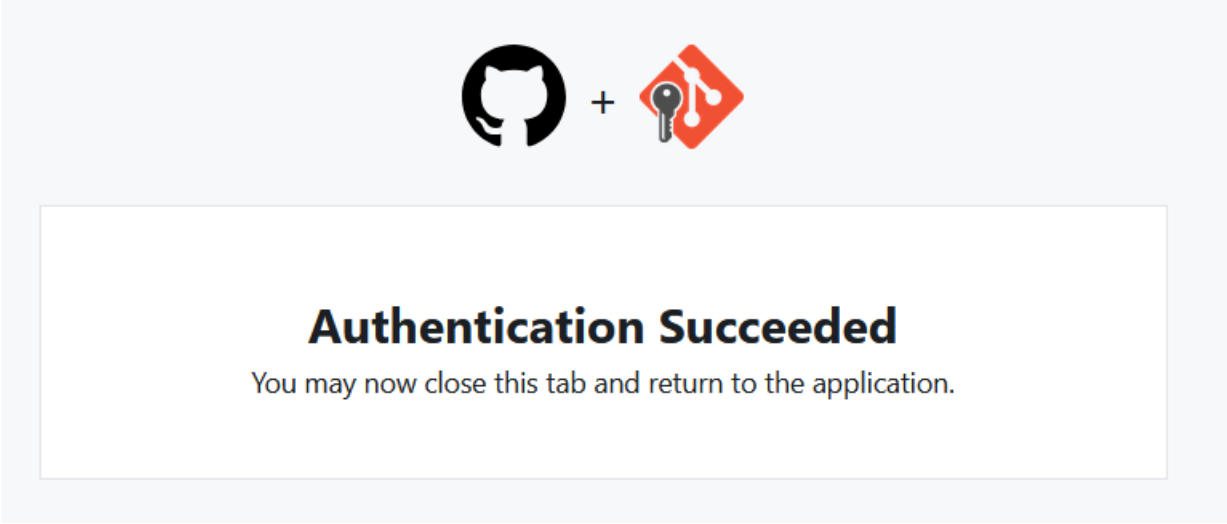
Note: If there is no existing project to clone, skip ahead to the section marked **Create a New Platform IO Project in VS Code**

1. Close all folders and projects in VS Code. **File/Close Folder** or **File/Close Workspace** will accomplish this.
2. **Browse** to the GitHub project you want to clone.
3. To clone the repository **click the green Code button** and **press the copy icon next to the URL** to copy the URL to the clipboard.
4. **Go to VS Code and click the Platform IO alien logo** in the left toolbar and choose **Clone Git Project** from the Quick Access pane.
5. In the window that opens at the top center of VS Code, press **CTRL-V** to paste the URL of the project and then press **ENTER**
6. A dialog opens asking where on the local machine to put the folder that contains the project. It's asking what folder it should put the folder that contains the local copy of the project. The answer to this question is always the folder **User Documents\PlatformIO\Projects**
7. **Highlight the Projects folder**, don't browse into that folder
8. **Click the Select as Repository Destination folder**
9. **If you don't have cached credentials for GitHub, you will be prompted to authenticate with your GitHub credentials:**

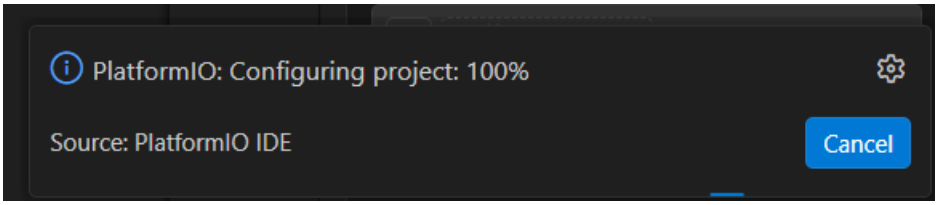


10. Click Sign in with your browser

11. This window will pop up saying Authentication Succeeded



12. The project will be downloaded to the local machine and open in VS Code.
13. Platform IO will download the managed libraries and install a compiler for this CPU type. This may take some time, and looks like this:



14. Wait for all tasks to complete before proceeding!

Create a New Platform IO Project in VS Code

Note: If you cloned an existing project earlier, you do not need to create a new project. Skip ahead to the section **Compiling and Uploading to the Microcontroller**

1. **Click the alien head Platform IO logo**  from the left-hand toolbar.
2. From the Quick Access menu (bottom left), choose **Projects & Configuration**
   Click the **+ Create New Project** button
3. **Choose a name**. Choose anything. This is temporary. The project name will be chosen later.
4. **Choose the development board** from the list which most closely matches the one we're working with.
5. Choose the Framework **Arduino**
6. Check the box that says **Location: Use Default Location** This directory is **User Documents\PlatformIO**
7. Click the **Finish** button and let it build the project.
8. Close the temporary project by choosing **File/Close Folder**
9. Browse to **User Documents\PlatformIO** and in that folder create the folder "Projects".
10. Go to Github.com and login with the user ID of the person where the project will be stored.
11. Choose **Repositories** from the top menu.
12. Click **+New**
13. Give the repository a name. Note if the name contains spaces, GitHub will change the name of the project.
14. Enter a description.
15. Set visibility to Private so we aren't republishing the code we're learning with.
16. Change the **Add README** slider to **On**
17. Click the green **Create Repository** button
18. From the page GitHub creates for the repository, **click the green <>Code** button.
19. Copy the URL of the project to the clipboard using **CTRL-C**
20. **Click the alien head / Platform IO logo** in the left menu bar.
21. From the Quick Access menu, click **Clone Git Project**
22. Paste the project URL on the clipboard to the field at the top of the VS Code project using **CTRL-V** and then press Enter.
23. A dialog opens. It's asking where you want to put the local copy of the repository. The answer is always **User Documents\PlatformIO\Projects** DO NOT browse into the Projects folder and then click OK. Rather **highlight the Projects folder and click OK**
24. A folder named with the project name will be created in the folder **User Documents\PlatformIO\Projects**

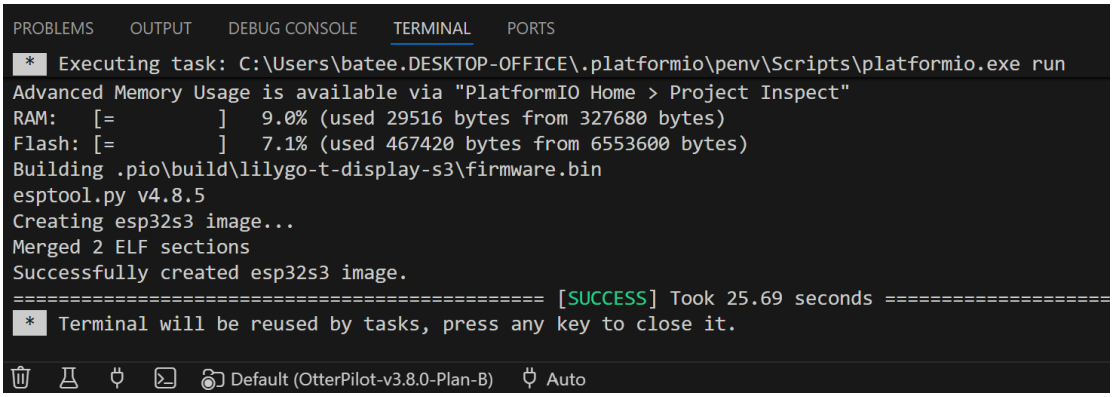
25. When the repository in GitHub has been cloned to the local machine, VS Code will prompt you to open the project. Always choose **In A New Window** as we don't know what changes have been made to the current window
26. Go to the temporary project folder that Platform IO created: **User Documents\PlatformIO\Projects\Temporary**
27. Copy all the files in that project to the clipboard using **CTRL-C**
28. Go to the folder that Platform IO created for the GitHub-linked project and paste them using **CTRL-V**
29. In VS Code, **click the Source Control icon**  from the left-hand toolbar.
30. We see a list of files which have been added/removed/changed since the last time we synched with GitHub.
31. Click the **(checkmark) Commit** blue button at the top left under Changes.
32. We get a "There are no staged changes to commit" message. Click **Yes** to stage all changes to commit.
33. A file called "COMMIT_EDITMSG" is created. It's asking us to tell GitHub what we did this time. Type **Added Initial Project Files** to the first line.
34. Close this file by **clicking on the Dot or X in the upper RH corner of the file name at the top of the file.**
35. **Yes** to Save.
36. A blue **Sync Changes** button appears in the top left corner of the Changes window. **Click the blue Sync Changes button** to sync with GitHub.
37. A message appears about pushing and pulling from GitHub. **Click Yes**
38. A new entry should appear in the GRAPH pane below the Changes pane, showing the commit we just made.
39. Go to the GitHub page for the repository we just created and refresh that page in the web browser.
40. We should see the folder structure of our project, along with the platformio.ini file that tells Platform IO how to compile our project.
41. Do we want VS Code to run git fetch periodically? Only really needed if someone else is working on the project with us, so let's **say no** to avoid confusion.
42. Go back to VS Code**. Click the Explorer icon**  at the top left of the left toolbar. This shows us the files in the local project.
43. **Create a file** called **/src/main.cpp** It should be of the same structure as an Arduino .ino file, with a setup function that runs once and a loop function that runs forever. A great source for examples would be the example code provided by the manufacturer with the development board or sensor.
44. Create any other files that your project needs.

45. If libraries (shared code) are supplied by the microcontroller or the devices the controller is connecting to, **click the Platform IO icon**  in the left toolbar and select **Libraries** from the Quick Access pane.
46. Select the library that supports your hardware and click it to browse to that library.
47. Click the **Add to Project** button
48. Select our project from the list and click “Add” to download it to the managed libraries location **/.pio/libdeps**
49. If a library is not available via Platform IO download, you can manually add that library to the unmanaged libraries folder **/lib**

Compiling and Uploading to the Microcontroller

Now that we have a project,

1. Connect a USB cable from the computer that VS Code is running on to the microcontroller.
2. In VS Code, **click the Platform IO Upload icon** in the bottom toolbar. The hover text will be Platform IO: Upload.
3. This takes the C source code and compiles it into machine code for the specific microcontroller configured in the project. Then VS Code will transfer that machine code to the microcontroller – this is called uploading.
4. When it's ready to upload, we see lines like "Connecting…" and "Writing…". Wait for the line that says SUCCESS in green text before proceeding:



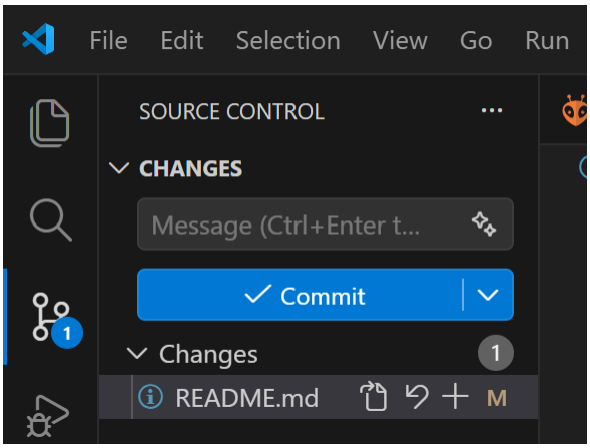
5. When finished, the microcontroller will reboot and start running the program uploaded to it.

Committing Updates to the Repository

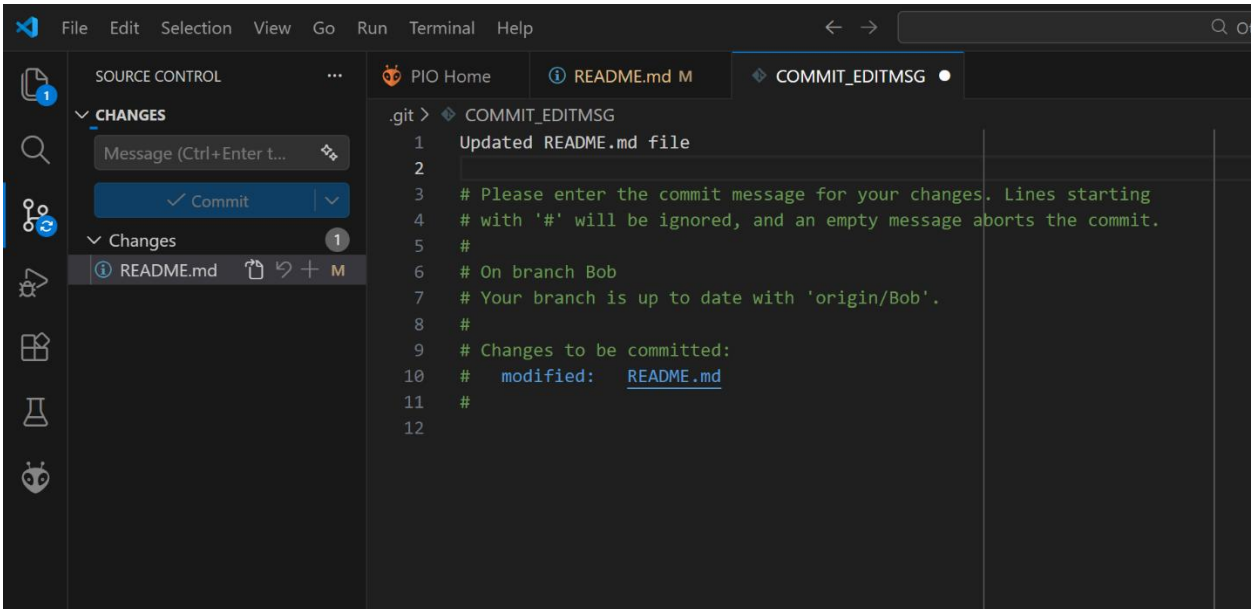
Committing to the repository is the opposite of checking out or cloning the repository. We are taking updates which were made on the local machine and uploading them to the repository.

Once we have made some update to the code on the local machine and want to store that code safely and share with others on the team, we can commit that code to the repository. This process will commit to the selected branch of the currently open project repository, as that's the branch where the changes were done.

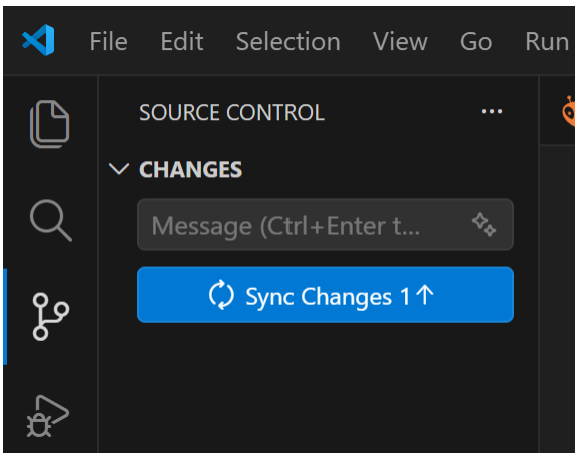
1. Save the work by clicking **File/Save** This saves each open and changed file in VS Code.
2. On the left toolbar **click the Source Control icon**: 
3. The blue dot with a numeral 1 means that one file on the local machine project is different from the repository. We see a list of those file(s) in the left pane of VS Code:



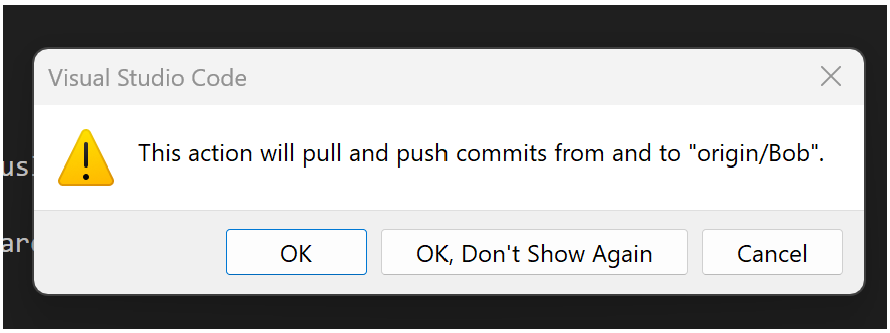
4. **Click the blue Commit button** to commit those changes to the repository. A new file called COMMIT_EDITMSG opens in the middle pane of VS Code:



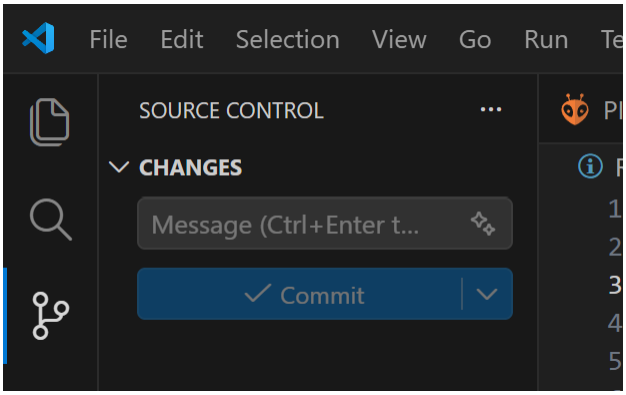
5. In the top line or lines, type what changed. **Updated README.md file** and any other helpful info, like **This isn't working** or **Jeremiah is a goof**
6. **Hover the mouse pointer over the white dot next to its name** and it will change to an X. Then **click Save** and the COMMITMSG file saves and closes.
7. After we commit the changes we made, we need to sync them to the repository. The blue Commit button has changes to a "Sync Changes" button. **Click the blue Sync Changes** button to upload the changes to the repository:



8. The Push/Pull dialog comes up, saying that it intends to download any changes that have occurred on the repository to the machine, and to upload any changes to the repository, with the goal to make the two the same. **Click OK**



9. Now the Source Control dialog shows no changes needed. The updates we made have been uploaded to the repository:

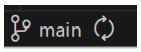


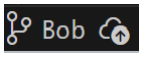
Branches

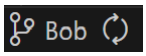
Branches are a way of creating different versions of code in a single project repository. Ideas are Production, Old, Development, etc. Branching is different than Forking, which would be to copy the repository contents wholesale to project repository with a different name.

Creating a New Branch

By creating a new branch, we're saying that we want to "branch" the project. This means take whatever code is open in the editor, which was a certain branch, and copy it to another branch with a new name. This leaves the current branch as-is and causes all updates after the branch to go to the new branch name. To create a new branch requires write permissions in the project:

1. Look at the Current Branch icon 

in the bottom left corner of VS Code. This shows that the current branch of the current project is **main**.
2. **Click the Current Branch icon** This will show a list of existing branches in the top center of VS Code.
3. If one of the branches shown is the one to be changed, go ahead and select it. If it doesn't exist, select **+Create a New Branch…** from that drop-down list in the top center of VS Code and enter the branch name and press Enter.
4. Now if we look in the bottom left corner, we see the branch has changed: 

5. There is a cloud icon with up arrow next to the name to indicate the new branch hasn't been published to the repository yet.
6. **Click the cloud icon to the right of the branch name** to publish the branch to the repository. The branch has been published to GitHub. We can verify that by going to the GitHub repository and looking at the list of branches.
7. Now the branch name appears like this: 

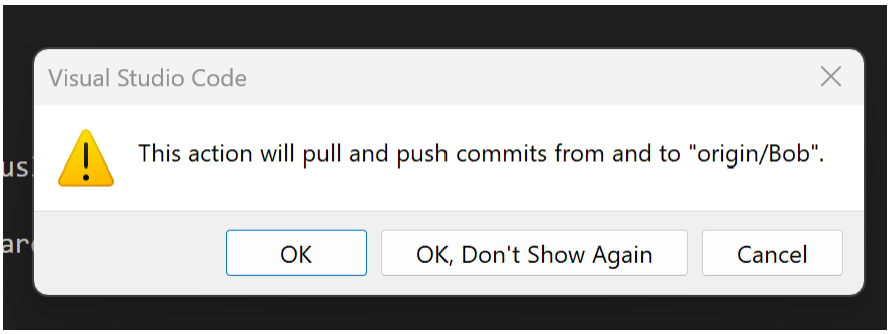
8. The new branch has been published, and the local project is synched with that new branch.

Switching Branches

Warning! This will pull the contents of the new branch into the project folder on the local machine **overwriting everything currently in that folder**!

Maybe we want to put this branch on hold and go back to an older branch.

1. Make sure to save the current project. Choose **File/Save** and optionally commit that work to the GitHub repository.
2. **Click the current branch name** in the bottom left corner of VS Code. **Select the desired branch** from the list of existing branches which appears at the top center of the window.
3. **Press the recycle icon** next to the name of the branch.
4. The push/pull dialog appears, telling us that it's going to sync with the new branch:



5. **Click OK**
6. When the updates finish, the new branch name will appear in the bottom left corner.

Scrapping Changes and Starting Over

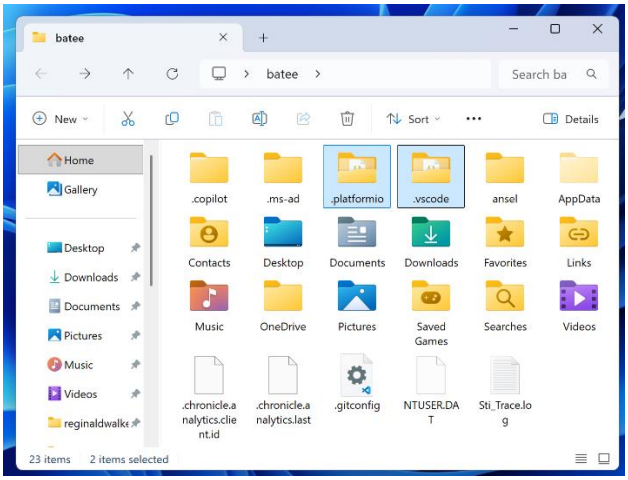
If things aren't working out, don't commit it to the repository! To start over:

1. Close VS Code folder or workspace with **File/Close Folder** or **File/Close Workspace**
2. Go to **User Documents\PlatformIO\Projects**
3. Rename the folder so that we can re-download the folder from the project. It will have the name of the repository. Or if it is no longer needed, just delete it
4. Go to the section marked **The Platform IO Project** and create it again or re-clone the existing project to the local machine.

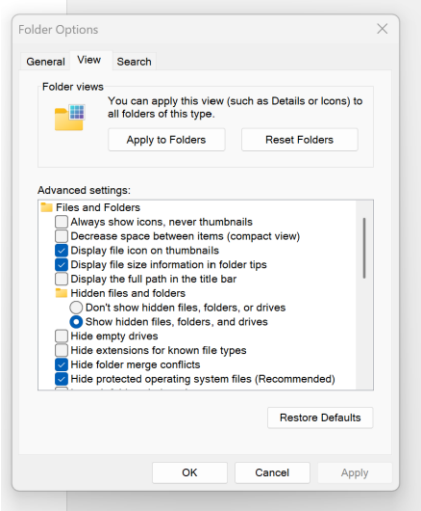
Things Still Aren't Working

It happens. The changes we're making with AI can mess something up in the VS Code installation, causing us to have to reinstall VS Code and its extensions.

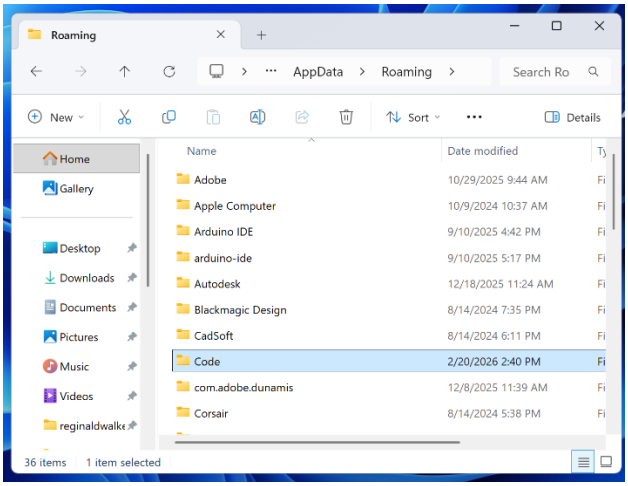
1. Close VS Code.
2. Go to the Start Menu and **click Settings** and then **click Apps** and then **Installed Apps**.
3. Find Microsoft Visual Studio Code in the list and uninstall it.
4. Find and Uninstall Git.
5. Go to Control Panel / Credential Manager. Click Windows Credentials. Find Git:https://github.com and remove it. Find "vscodevscode.github-authentication…" and remove it.
6. Once the uninstaller completes, go to the **User\Documents** folder:



7. **Highlight the folders .platformio** and **.vscode**. If they don't appear in the list of folders, it may be necessary to click "… and then Options and then View to see Folder Options:



8. **Check Hidden files, folders, or drives** and then **click OK**
9. Delete the folders marked **.platformio** and **.vscode**.
10. Browse into the **UserID\App Data** folder, and then the Roaming folder:



11. **Highlight the folder Code** and **delete it**
12. Reboot Windows to clear any pending file and folder deletions.
13. Go to the section marked **Setting Up a New Computer** and follow the instructions to reinstall and reconfigure VS Code.